Universidad Autónoma de Madrid

Escuela politécnica superior

Doble Grado en Ingeniería Informática y Matemáticas

Trabajo Fin de Grado

SIMILITUDES BASADAS EN MÉTRICAS NO EUCLÍDEAS

Autor:Daniel Redondo Castilla

Tutor: Fernando Díez Rubio

Junio 2017

SIMILITUDES BASADAS EN MÉTRICAS NO EUCLÍDEAS

Autor:Daniel Redondo Castilla
Tutor: Fernando Díez Rubio

Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio 2017

Resumen

Resumen

Los sistemas de recomendación son una herramienta ampliamente utilizada hoy en día. Una de las aproximaciones que goza de mayor popularidad es el filtrado colaborativo, que trata de buscar usuarios similares entre sí para realizar las recomendaciones. Nuestro objetivo es analizar una nueva forma de calcular las similitudes entre usuarios, utilizando la distancia de Mahalanobis, y comparar sus resultados con las medidas de similitud clásicas. Para ello, hemos investigado en la literatura aplicaciones de la distancia de Mahalanobis en distintos ámbitos. Posteriormente, hemos diseñado un sistema para realizar estos experimentos, considerando de manera especial la forma de implementar la distancia de Mahalanobis adaptada a este problema. Para probar este nuevo sistema, se han utilizado los datos de valoraciones de películas MovieLens-100K. Tras analizar los resultados, concluimos que la nueva aproximación no mejora las medidas de similitud consolidadas actualmente para sistemas de recomendación por filtrado colaborativo. Sin embargo, se proponen ciertas nuevas aproximaciones que podrían mejorar los resultados obtenidos.

Palabras Clave

Sistemas de Recomendación, Distancia de Mahalanobis, Filtrado Colaborativo, Similitud

Abstract

Recommender systems are a widely used tool nowadays. One possible approach are collaborative filtering systems, which try to find similarities between users for the purpose of recommending. Our goal is to investigate a new way to compute user similarities, using the Mahalanobis distance, and compare it to the traditional metrics used for this purpose on recommender systems. To do so, we first investigated various applications of the Mahalanobis distance in different areas. Secondly, we designed a system to carry out the experiments, especially considering the way to implement the Mahalanobis distance applied to our problem. In order to test this system, we used the Movielens-100K movie ratings database. After analyzing the results obtained, we concluded that this new approach does not improve the traditional similarity measures employed in recommender systems. However, we propose different techniques that could improve this results.

Key words

Recommender Systems, Mahalanobis Distance, Collaborative Filtering, Similarity

Agradecimientos

A Fernando, por guiarme en este camino.

A mis amigos, por su apoyo incondicional.

A Ana, porque sin ella no habría llegado hasta aquí.

A mis padres, por educarme como lo hicieron.

Índice general

Índice de Figuras

1 Introducción

1.1. Motivación del proyecto

Un sistema de recomendación es un conjunto de herramientas implementadas en un sistema software destinadas a proporcionar a un cliente una recomendación relativa a los objetos ofrecidos por el sistema, como productos, música, noticias, etc. que podrían resultar interesantes para consumo de los usuarios. Se trata de una herramienta con gran expansión actual, ya que además de ser utilizada en multitud de servicios que todo el mundo conoce, como *YouTube*, con sus vídeos recomendados para cada usuario, o *Amazon*, con sus recomendaciones de productos según el historial de búsqueda y compras, también tiene detrás un gran interés científico, ya que hay muchos grupos de investigación, tanto públicos como privados, tratando de mejorar y hacer más precisas las recomendaciones en un intento de comprender y modelar los comportamientos de los usuarios.

La utilidad de estos sistemas con respecto a los usuarios es clara: la posibilidad de obtener resultados de manera más sencilla, sin tener que navegar por la inmensa cantidad de información del sistema, mejora y agiliza en gran medida la experiencia de usuario, lo que en definitiva puede reportar un mayor beneficio a las empresas.

Existen varias aproximaciones a la hora de implementar un sistema de recomendación, entre las cuales una de las más populares es la denominada por *filtrado colaborativo*, en el que, para generar resultados, usamos los datos de los demás usuarios, buscando relaciones entre éstos y el usuario al que se proporciona la recomendación. Una forma de implementar un sistema basado en filtrado colaborativo consiste en realizar un *sistema de recomendación basado en vecindario*, en el cual intentamos encontrar los usuarios más similares al que estamos recomendando y basarnos en sus gustos a la hora de recomendar. Sin embargo, cómo caracterizamos los usuarios más similares no es una cuestión trivial.

Tradicionalmente se han utilizado aproximaciones basadas en métricas euclídeas, como la similitud basada en el coseno.

Por otro lado, las distancias estadísticas, que son aquellas que cuantifican las diferencias entre dos objetos estadísticos, como variables aleatorias o distribuciones de probabilidad, tienen numerosas aplicaciones en campos como la biología, genética, psicología, etc. Esto es debido, entre otras cosas, a que utilizan información presente en el resto de datos a la hora de calcular la distancia entre dos objetos, no sólo datos de los objetos en cuestión.

A la vista de esto, cabe preguntarse si la aplicación de alguna de estas distancias puede mejorar el cálculo de la similitud entre usuarios en los sistemas de recomendación, con el fin último de mejorar las recomendaciones. Puesto que la respuesta a esta pregunta requeriría cierto grado de experimentación, consideramos que resultaría interesante su estudio.

1.2. Objetivos y enfoque

Como consecuencia de lo anterior, el objetivo general de este trabajo consiste en profundizar en los conceptos subyacentes a los sistemas de recomendación, en concreto el relativo al cálculo de similitudes, para aprender a utilizar estos sistemas e intentar mejorar los resultados de los métodos estándar para el cálculo de similitudes descritos en la literatura. Para llevar esto a cabo, definimos los siguientes objetivos:

- Buscar en la literatura aplicaciones de distancias estadísticas en diversos campos y determinar si alguno es aplicable a la hora de realizar un sistema de recomendación.
- Analizar las distintas aproximaciones posibles para implementar un sistema de recomendación y elegir la más propicia para usar una distancia estadística.
- Analizar distintas herramientas que puedan ser útiles para realizar un sistema de recomendación con las características requeridas o, en caso de no encontrarlas, diseñar una herramienta propia para posteriormente implementarla.
- Analizar los resultados obtenidos para comprobar si la aproximación tomada mejora el rendimiento con respecto a las aproximaciones clásicas.

Para llevar a cabo este trabajo, hemos necesitado conocimientos adquiridos en las siguientes asignaturas cursadas en el Doble Grado:

- **Álgebra Lineal:** han sido necesarios conocimientos sobre manipulación de matrices y vectores.
- **Programación I y II:** en estas asignaturas aprendimos los conceptos básicos de la programación, además del uso de ciertas estructuras de datos que hemos utilizado en este trabajo.

- **Laboratorio:** en esta asignatura utilizamos por primera vez el lenguaje *Python* aplicado al procesamiento de datos matemáticos con la librería *Sage*
- **Análisis y diseño de software:** en ella vimos por primera vez el paradigma de la programación orientada a objetos, además de aprender diversas herramientas para ayudar en el diseño de software, como los diagramas de clases y secuencia
- **Análisis de Algoritmos:** en ella estudiamos la implementación de diversos algoritmos, así como formas de medir su eficiencia
- **Estadística I y II:** en estas asignaturas estudiamos los conceptos de variables aleatorias, covarianzas y la distancia de Mahalanobis, que han sido necesarios para la realización de este trabajo.
- **Inteligencia Artificial:** en esta asignatura estudiamos como entrenar modelos a partir de conjuntos de entrenamiento y test.

2

Estado del arte

El desarrollo de tecnologías de la información para los sistemas de recomendación ha evolucionado enormemente durante las últimas dos décadas: desde los trabajos iniciales de [1] y [2], que establecieron las bases de los sistemas de recomendación (en concreto, por filtrado colaborativo), hasta la expansión actual en múltiples revistas, conferencias y, por supuesto, modelos de negocio llevados adelante por empresas del sector de las TIC, como *Google* o *Amazon*.

En particular en el presente trabajo queremos profundizar, como hemos expresado en la introducción, en la obtención de vecindarios de un cliente mediante el cálculo de similitudes de usuarios del sistema relacionados con el cliente. La diferencia respecto de otros trabajos previos radica en que basamos el cálculo de la similitud en el uso de distancias estadísticas, como la distancia de Mahalanobis.

Esta distancia se ha empleado en problemas que pueden guardar relación con el trabajo realizado, como por ejemplo en [3]. En este trabajo, la distancia se emplea para parametrizar una transformación lineal cuyo objetivo es separar los datos que pertenecen a distintas clases y agrupar los que pertenezcan a la misma, de manera que sea más eficaz aplicar un algoritmo de clasificación de patrones por k vecinos próximos.

La distancia de Mahalanobis también ha sido empleada en otros ámbitos, por ejemplo en [4], donde los autores proponen un nuevo modelo para agrupar zonas hidrológicas de acuerdo a su frecuencia de que ocurran inundaciones, estableciendo su grado de similitud de acuerdo al radio a partir del cual se intersecan las elipses equidistantes de acuerdo a la distancia de Mahalanobis. Finalmente, en [5] se utiliza una distancia similar a la de Mahalanobis para realizar recomendaciones de música usando datos contextuales y votos de los usuarios, pero tomando la matriz de covarianzas como una variable a aprender en el modelo.

Como hemos establecido en la introducción, comenzamos el trabajo mediante la investigación de aplicaciones de distintas distancias estadísticas en varios campos científicos. Tras esta fase inicial, procedimos a investigar las distintas aproximaciones que se pueden tomar a la hora de implementar un sistema de recomendación. Para ello, recurrimos al libro *Recommender Systems Handbook* [6], donde vienen descritas con un buen nivel de detalle las distintas formas de recomendar que se pueden utilizar. Nos centramos mayoritariamente en métodos que pudieran usar un factor de similitud entre objetos o usuarios, como aquellos basados en vecindarios, debido al interés del trabajo por usar tipos de distancia especiales a la hora de calcular similitudes.

Con el desarrollo de los sistemas de recomendación que acabamos de mencionar, ha venido aparejada la publicación de aplicaciones y herramientas específicas tanto para la investigación en este área como para el desarrollo de productos y servicios. Entre las más destacadas hemos de mencionar *MyMediaLite* [7], desarrollada en 2011 por Zeno Gantner, Steffen Rendle, Lucas Drumond y Christoph Freudenthaler en la Universidad de Hildesheim. Esta librería proporciona una gran cantidad de herramientas para realizar recomendaciones y experimentos basados en recomendadores de forma sencilla: desde varios tipos de recomendación (predicciones de ratings numéricos o predicción de objetos basado en datos unarios como clicks o compras), pasando por numerosos algoritmos de recomendación personalizables con distintos parámetros (como filtrado colaborativo o modelos de factorización de matrices), y por último varias medidas de error para cuantificar la calidad de las recomendaciones (como RMSE, MAE o nDCG). Además, presenta un modo de uso sencillo mediante interfaz de línea de comandos, junto a la posibilidad de incorporarlo a programas propios mediante una API. Por último, se distribuye bajo una licencia GPL, lo que facilita su uso y distribución.

Por otra parte, mencionamos la librería *Mahout* [8], que comprende una gran cantidad de algoritmos relacionados con el aprendizaje automático, en concreto, algoritmos de clustering, de clasificación de patrones y de filtrado colaborativo. *Mahout* es utilizada por gran cantidad de grupos de investigación, grupos docentes y proyectos comerciales, gozando de una gran popularidad. *Mahout* forma parte de la *Apache Software Foundation*, y usa para gran cantidad de sus algoritmos la librería de computación distribuida *Hadoop*.

3

Conceptos teóricos

En esta sección se presentan los conceptos necesarios para comprender el contexto en el que se realiza el trabajo, entre los que se encuentran nociones básicas sobre los sistemas de recomendación y el concepto de distancia, y su relación entre ellos.

3.1. Sistemas de recomendación

Un sistema de recomendación comprende el uso de herramientas software y técnicas de análisis de datos con el objetivo de proporcionar sugerencias sobre objetos que podrían ser útiles a los usuarios de un sistema. Aquí, la palabra "objeto" es deliberadamente genérica, y cuando hablamos de ellos nos referimos a lo que el sistema recomienda a los usuarios, sean películas, vídeos, artículos del hogar, noticias, etc. Los sistemas de recomendación utilizan diversos diseños, interfaces gráficas y técnicas de recomendación, de manera personalizada, para obtener sugerencias efectivas y que satisfagan a los usuarios. Por otra parte, en el lado del usuario un sistema de recomendación es útil para navegar por la potencialmente abrumadora cantidad de datos del sistema y elegir objetos acordes con sus gustos de manera más sencilla.

Los sistemas de recomendación habitualmente se clasifican en (ver [9]):

- **Por popularidad:** Con esta aproximación trivial a los sistemas de recomendación, simplemente se recomienda al usuario los objetos más populares actualmente. Este sistema está lejos de ser óptimo, debido a la ausencia total de personalización. Sin embargo, es una técnica a tener en cuenta para combinarla con sistemas que tengan problemas con el *arranque en frío*, es decir, la capacidad de recomendar cuando hay pocos datos en el sistema.

- **Basados en contenido:** Esta aproximación se basa en emplear las características comunes de los objetos que han recibido una valoración favorable por parte del usuario activo para recomendarle nuevos objetos. La ventaja de este sistema es su gran precisión a la hora de recomendar, hecho que lo hace una alternativa muy popular a la hora de implementar un sistema de recomendación, como por ejemplo, en la plataforma de vídeos *YouTube*. Sin embargo, este sistema presenta el problema de la *sobre-especialización*, ya que sólo tiene en cuenta objetos que ha valorado el usuario y, por tanto, puede que nunca recomiende objetos algo distintos a lo que el usuario está habituado pero que le resultarían interesantes de todas maneras.

- **Filtrado colaborativo:** En esta aproximación utilizamos la intuición de que al usuario activo le parecerán interesantes objetos que hayan recibido una buena valoración por parte de usuarios con gustos similares a los suyos. De esta forma, utilizaremos datos de todos los usuarios del sistema a la hora de hacer recomendaciones. Puesto que es un sistema interesante que merece una mayor atención, ya que va a ser la aproximación tomada para nuestro proyecto, expandiremos esta idea en el apartado siguiente.

- **Demográficos:** Estos sistemas utilizan datos demográficos de los usuarios, como por ejemplo, la edad o el idioma, para realizar las recomendaciones. El ejemplo más simple de este tipo de sistemas está en las páginas web presentadas en el idioma principal del país desde el que se está accediendo.

- **Basados en conocimiento:** Esta aproximación se basa en tener un conocimiento intrínseco del problema, de manera que la recomendación se basa en saber cómo de útil será un objeto para el usuario. Para ello, se realiza una correspondencia entre las necesidades del usuario, que deben ser especificadas lo más claramente posible, y los objetos que cumplen dichas necesidades. Este tipo de sistemas necesita un gran esfuerzo para modelar el problema concreto, lo cual compensa la relativa poca cantidad de proceso que debe hacer el sistema para cada recomendación.

- **Basados en comunidad:** Esta aproximación utiliza las preferencias de los amigos de los usuarios a la hora de realizar recomendaciones. Este tipo de sistema, basado en la idea de que los usuarios confían más en las recomendaciones realizadas por sus amigos que en las de potenciales desconocidos, necesita una infraestructura de red social para poder utilizarse. Por ello, es ideal para ser utilizado en redes sociales ya existentes como Facebook, y proporciona recomendaciones de tal forma que es sencillo ver su procedencia.

- **Sistemas híbridos:** Un sistema de recomendación híbrido combina alguna de las técnicas vistas anteriormente para realizar su función. Por ejemplo, un sistema de filtrado colaborativo podría usar la aproximación por popularidad para usuarios que han valorado pocos objetos, o cuando el sistema es reciente y aún no tiene demasiadas valoraciones.

3.1.1. Filtrado colaborativo

Como hemos especificado anteriormente, una de las técnicas empleadas para los sistemas de recomendación es utilizar datos de los distintos usuarios para las recomendaciones al usuario activo, es decir, el que está solicitando la recomendación. En concreto, se usa la información de los usuarios con gustos más parecidos al usuario activo (por ejemplo, usuarios que tienen puntuaciones similares en varias películas, o que han mirado las mismas noticias), recomendando a dicho usuario objetos que también sean interesantes para estos usuarios activos. Uno de los problemas a los que se enfrenta esta aproximación se encuentra en el *arranque en frío*, mencionado anteriormente. Sin suficientes datos sobre los demás usuarios, los cálculos de similitudes entre distintos usuarios serán imprecisos cuanto menos, y pueden dar lugar a recomendaciones erróneas e insatisfactorias. Otro problema al que se enfrenta un sistema de recomendación por filtrado colaborativo es la potencial cantidad de datos que debe procesar para realizar su función adecuadamente. En una base de datos con millones de usuarios y objetos, el coste computacional asociado al procesamiento de todos esos datos puede ser prohibitivo e inmanejable para proporcionar recomendaciones en tiempo real. Para solucionar este problema, podemos optar por reducir el conjunto de datos de acuerdo a ciertos criterios para intentar perder la menor cantidad de información posible. Por otro lado, estos sistemas presentan grandes ventajas que los convierten en la opción predilecta en algunos casos. Podemos destacar la potencial precisión de estos sistemas. Según aumenta la cantidad de datos proporcionados por los usuarios, la información relativa a todo el sistema será más precisa y por tanto podremos recomendar con más exactitud a los usuarios. También remarcamos la capacidad de estos sistemas para evitar el problema de la *sobre-especialización*, pudiendo recomendar objetos que potencialmente gustarán al usuario pero que no sean directamente similares a los objetos que valora positivamente. Este hecho se conoce como *serendipia*, y ayuda a los usuarios a descubrir objetos que de otra forma posiblemente no hubiera visto nunca.

Entre los sistemas de recomendación por filtrado colaborativo distinguimos:

- **Sistemas basados en modelos:** En estos sistemas, se usan los datos del sistema para entrenar un modelo de datos que intente predecir las relaciones entre usuarios y objetos. Existen varias aproximaciones para este sistema, como los modelos matriciales o las redes neuronales
- **Sistemas basados en vecindario:** En estos sistemas, los datos se emplean directamente para realizar recomendaciones. En la siguiente sección profundizaremos sobre los detalles de este tipo de sistemas y las ventajas que presentan.

3.1.2. Sistema de recomendación basado en vecindarios

En un sistema de recomendación basado en vecindarios, los datos proporcionados por el sistema, o filtrados anteriormente, se utilizan para calcular los usuarios con gustos más parecidos al usuario activo, llamados *vecinos*, y éstos se usan para predecir la valoración que el usuario hará de un objeto aún sin puntuar (Figura 3.1).

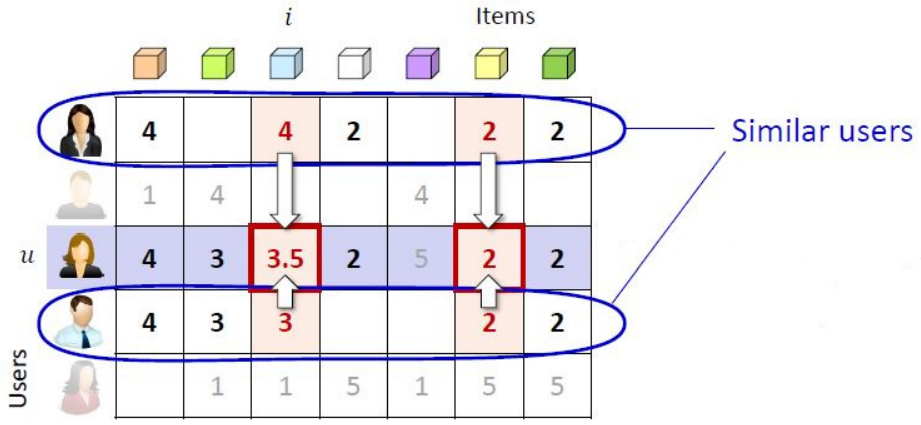

Figura 3.1: Vecinos más próximos [1]

Los sistemas de recomendación basados en vecindarios presentan varias ventajas que los destacan como una opción interesante a la hora de implementarlos. Por ejemplo, la sencillez a la hora de entenderlos e implementarlos, la capacidad de justificar las predicciones realizadas de manera sencilla, lo cual es importante en ciertos casos para aumentar la satisfacción del usuario; su mayor eficiencia frente al coste computacional de entrenar modelos complejos, su capacidad de proporcionar recomendaciones novedosas para el usuario y su estabilidad frente a la adición de nuevos usuarios, objetos o valoraciones, ya que sólo habría que volver a computar la similitud entre este nuevo usuario o el que ha sido modificado y los demás.

3.2. Herramientas matemáticas

Definición: Sea $X = x_1, x_2, ..., , x_n$ un conjunto finito. Su *cardinal* $|X|$ se define como su número de elementos, es decir

$$|X| = n$$

3.2.1. Descomposición en Valores Singulares (SVD)

La descomposición en valores singulares es un método para factorizar matrices de cualquier dimensión.

Sea M una matriz de dimensión $n \times m$. Entonces existe una descomposición de dicha matriz de la forma:

$$M = U\Sigma V^t$$

De tal forma que:

- U es una matriz *ortogonal* de dimensión $n \times n$

[1] Fuente: Apuntes de la asignatura *Búsqueda y Minería de Información*, UAM

- V es una matriz *ortogonal* de dimensión $m \times m$
- Σ es una matriz de dimensión $n \times m$ tal que en la diagonal tiene elementos no negativos y en el resto de posiciones tiene 0

A los valores de la matriz Σ se les llama *valores singulares* de M

Aplicación: pseudoinversa de Moore-Penrose

La *pseudoinversa de Moore-Penrose* [10] de una matriz es una generalización del concepto de matriz inversa (esto es, una matriz para la cual $MM^{-1} = M^{-1}M = I$). Dada una matriz real M de dimensión $n \times m$, su pseudoinversa de Moore-Penrose es la matriz M^+ de dimensión $m \times n$ que cumple:

- $MM^+M = M$
- $M^+MM^+ = M^+$
- $(MM^+)^t = MM^+$
- $(M^+M)^t = M^+M$

Si la matriz es invertible, la pseudoinversa de Moore-Penrose será igual a la inversa.

La pseudoinversa de Moore-Penrose tiene la interesante propiedad de proporcionar el resultado que minimiza el error cuadrático de un sistema que no tiene solución. Es decir, dado un sistema $Mx = b$, el vector $x = M^+b$ nos da la solución con menor error cuadrático.

El método SVD visto nos proporciona una manera de calcular esta matriz: si la descomposición en valores singulares de una matriz M es

$$M = U\Sigma V^t$$

Entonces su pseudoinversa de Moore-Penrose será:

$$M^+ = V\Sigma^+U^t$$

Donde Σ^+ se obtiene de sustituir todos los elementos distintos de 0 de Σ por sus respectivos inversos multiplicativos (es decir, el elemento x^{-1} para el cual $x \cdot x^{-1} = 1$) y posteriormente trasponerla.

3.2.2. Distancia

En matemáticas, una distancia se define de la siguiente manera [11]:

Sea X un conjunto. Sea d una función tal que $d : X \times X \to \mathbb{R}$.

d es una *distancia* si cumple las siguientes propiedades:

1. $\forall a, b \in X,\ d(a,b) \geq 0$

2. $\forall a, b \in X,\ d(a,b) = d(b,a)$

3. $\forall a \in X,\ d(a,a) = 0$

4. $d(a,b) \leq d(a,c) + d(c,b),\ \forall a, b, c \in X$

5. $\forall a, b \in X,\ d(a,b) = 0 \Leftrightarrow a = b$

Si cumple las propiedades 1, 2 y 3 se dice que es una *disimilaridad*.

Si una *disimilaridad* también cumple 4 y 5, se le llama una *distancia métrica*.

Cuando hablamos de una *distancia*, formalmente nos referimos a una *distancia métrica*.

3.3. Distancia de Mahalanobis

La distancia de Mahalanobis [12] es una función que tiene en cuenta la covarianza del conjunto de datos a la hora de calcular la distancia entre dos puntos.

Sea X una matriz $(n \times d)$, que comprende n objetos medidos por d variables. Sean x_1, x_2 dos objetos de dicha matriz, es decir, dos filas.

La distancia de Mahalanobis entre dichos puntos se define como:

$$d(x_1, x_2) = \sqrt{(x_1 - x_2)^t \Sigma^{-1} (x_1 - x_2)}$$

Siendo Σ la *matriz de covarianzas* del conjunto de datos.

Se aprecia la similitud de esta distancia con la distancia euclídea:

$$d(x_1, x_2) = \sqrt{(x_1 - x_2)^t (x_1 - x_2)}$$

Siendo la distancia de Mahalanobis equivalente a la distancia euclídea para variables aleatorias con covarianza entre sí 0 y varianza 1 (lo que equivale a una matriz de covarianzas igual a la identidad).

En la figura 3.2 podemos ver la relación entre la distancia euclídea y la distancia de Mahalanobis. Las líneas representan puntos equidistantes respecto del centro de los datos según la distancia que se está analizando (a la izquierda, la distancia euclídea; a la derecha, la distancia de Mahalanobis).

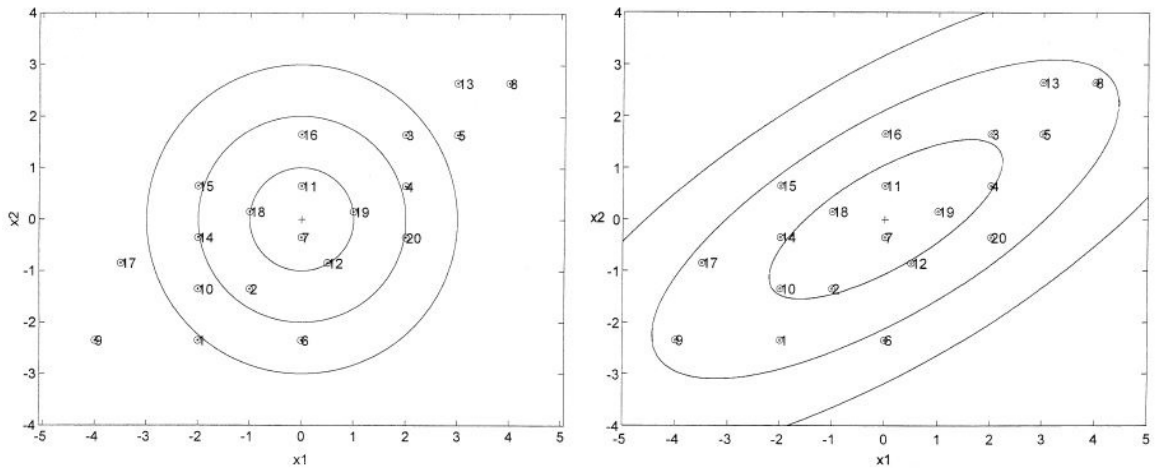

Figura 3.2: Distancia euclídea y Mahalanobis [2]

3.4. Relación entre las distancias y los sistemas de recomendación

Una vez realizada esta introducción, cabe preguntarse donde encajan las distancias dentro del uso de un sistema de recomendación. Como hemos visto anteriormente, los sistemas de recomendación basados en vecindario utilizan la similitud entre los distintos usuarios para realizar predicciones.

Utilizando el *feedback* que nos proporcionan los usuarios al utilizar el sistema, ya sea en forma de valoraciones explícitas como *likes* y puntuaciones, o implícitas como el historial de navegación, podemos identificar al usuario como un vector cuyas componentes sean estas valoraciones sobre los distintos objetos del sistema: la primera coordenada correspondería al objeto 1, la segunda al objeto 2, y así sucesivamente.

$$\begin{array}{cc} & \begin{array}{ccccc} i_1 & i_2 & i_3 & i_4 & i_5 \end{array} \\ U = & \begin{pmatrix} 2 & \_ & 5 & 1 & \_ \end{pmatrix} \end{array}$$

Figura 3.3: Usuario como vector de puntuaciones

Como observamos en la figura 3.3, no todos los objetos han de tener valoración por parte del usuario, lo que dificulta el problema del calculo de similitudes. Sin embargo, al tener que calcular la distancia entre dos usuarios, podemos utilizar únicamente las componentes de los vectores que tengan valor para ambos usuarios, es decir, los objetos que hayan sido valorados por los dos simultáneamente Utilizando esta aproximación, podemos emplear por ejemplo la distancia euclídea entre los vectores para calcular su similitud: cuanta mayor distancia, menos similares. Otra variante basada en ésta, más popular, es utilizar el coseno del ángulo que forman ambos vectores como medida de similitud: cuanto menor sea el ángulo, y por tanto mayor el coseno, más similares serán ambos usuarios (figura 3.4).

[2]Fuente: *The Mahalanobis Distance* [12]

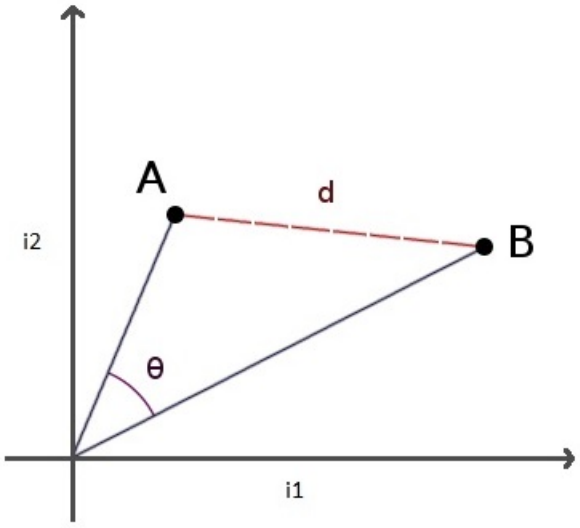

Figura 3.4: Distancia euclídea y coseno

Siguiendo con la aproximación de utilizar una distancia para calcular la similitud entre usuarios, proponemos explorar las posibles ventajas de utilizar distancias menos populares, o que explotan otras características de los objetos, como la distancia de Mahalanobis, a la hora de ver la proximidad entre usuarios. Esta distancia tiene en cuenta el conjunto de datos completo, concretamente la correlación entre las variables que lo componen, a la hora de realizar los cálculos, por tanto cabe pensar que podría modelar eficazmente las similitudes entre usuarios.

4

Sistema, diseño y desarrollo

4.1. Introducción

El uso de las librerías mencionadas en la Sección 2, algunas de las cuales no contemplan la implementación del cálculo de vecinos usando las distancias estadísticas descritas, nos sugirió la posibilidad de realizar implementaciones ad-hoc de las mismas, con el objeto de llegar a incluirlas, inclusive, como parte de las librería existentes.

Por tanto, se efectuó una implementación de la distancia de Mahalanobis en C#, utilizando la información proporcionada en [12]. La idea de esta implementación era incorporarla a la librería *MyMediaLite* para realizar los experimentos pertinentes. Sin embargo, tras un periodo de documentación sobre esta librería, decidimos rechazar esta idea, debido a la potencial complejidad que supondría realizar esta tarea. La compleja API de la librería, destinada a un uso más profesional y para experimentos con varios tipos de algoritmos y distintos tipos de datos, nos impedía ver el alcance total de la tarea. De igual modo, revisamos la librería *Mahout* para intentar realizar la implementación ahí, pero acabamos llegando a la misma conclusión.

De este modo, decidimos que la mejor opción, acorde con el alcance del trabajo, era implementar un sistema de recomendación propio, más sencillo y diseñado para la tarea concreta que queríamos realizar. Para realizar el diseño de la aplicación a desarrollar, estudiamos el recomendador *python-recsys*, por Óscar Celma, incorporando las estructuras básicas presentes en él pero adaptándolo a nuestras necesidades, y añadiendo la clase pertinente al algoritmo basado en vecindarios, que es el objetivo principal de este trabajo.

4.2. Diseño e implementación del sistema

Tal y como hemos explicado en la sección anterior, la decisión tomada para realizar los experimentos con las nuevas distancias consiste en implementar un sistema de recomendación propio, usando la aproximación basada en vecindarios.

En primer lugar realizamos un diseño del sistema mediante un diagrama de clases, con el objetivo de organizar las ideas y obtener un software modular y fácilmente ampliable en caso de querer realizar experimentos con más algoritmos y distancias o utilizar información contextual de los datos.

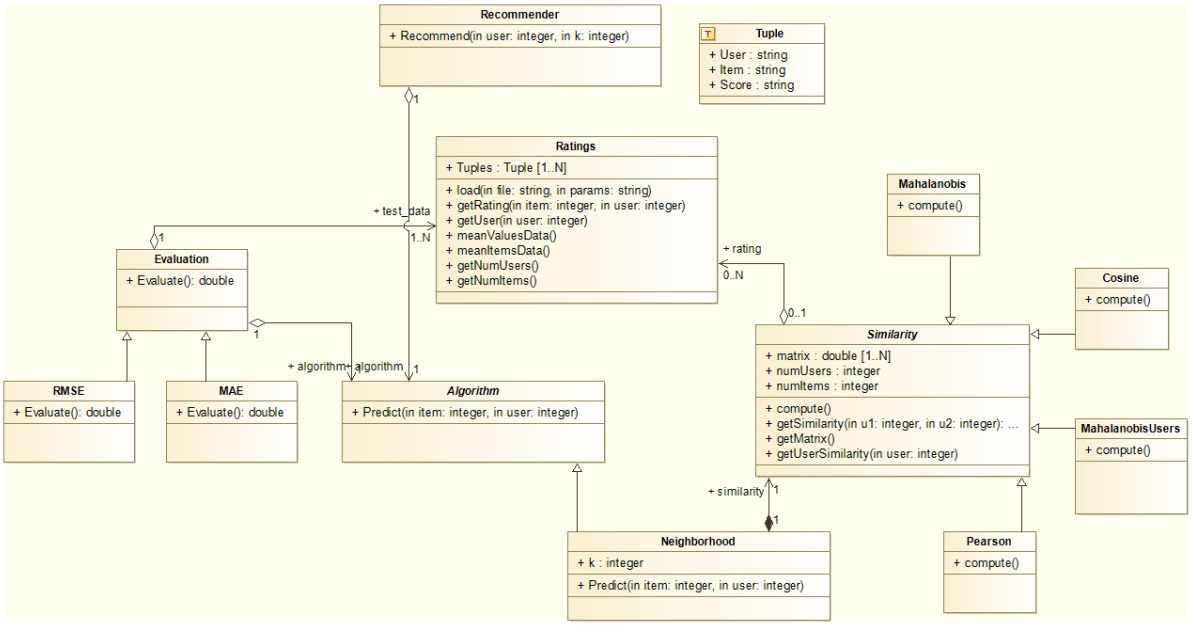

Figura 4.1: Diagrama de clases del sistema desarrollado

En la figura 4.1 podemos observar el diseño de clases del sistema desarrollado. Éste se ha orientado a realizar una aplicación que facilite la realización de experimentos, y no tanto como un sistema de recomendación útil para ser usado en un sistema real.

Hemos elegido *Python* como lenguaje para implementar el sistema propuesto. *Python* es un lenguaje de alto nivel y ampliamente utilizado, lo que facilita encontrar documentación y ayuda online para que sea más sencillo realizar la implementación. Además, *Python* cuenta con un potente y eficiente sistema de *comprensión de listas*, que facilita la operación con listas de elementos; y una librería para computación científica en *NumPy*, que ha sido utilizada para numerosas funciones en la implementación realizada, como las operaciones con vectores y matrices o el cálculo de la matriz de covarianzas.

La clase *Ratings* se encarga de guardar la matriz de valoraciones utilizada para realizar experimentos, junto con ciertos métodos orientados a proporcionar información a las otras clases de manera más sencilla. Para guardar los datos, se utiliza la clase *array* de la librería *NumPy*, ya que contiene gran cantidad de métodos destinados a facilitar el procesamiento de datos. La matriz contiene las valoraciones que los usuarios han dado

a los objetos, correspondiendo los índices de las filas a los usuarios y los índices de las columnas a los objetos.

De la clase abstracta *Algorithm* heredarán todos los algoritmos cuya función sea, dado un índice de usuario y un índice de objeto, predecir la valoración que dicho usuario le dará al objeto.

En este caso, hemos implementado un algoritmo basado en vecindarios, utilizando los k vecinos más próximos al usuario a la hora de realizar la predicción de la valoración. Para calcular la valoración del objeto que predecimos que dará el usuario, utilizamos la media ponderada de las valoraciones de los vecinos más próximos al usuario [13]:

Sea K_u el conjunto de los k usuarios más próximos al usuario activo. Sea r_{ui} la valoración dada por el usuario u al objeto i. Sea s_{uv} el coeficiente de similitud entre los usuarios u y v. Mediante el método basado en vecindarios, predecimos la valoración que dará el usuario, $\hat{r}_{ui}$ de la siguiente forma:

$$\hat{r}_{ui} = \frac{1}{R} \sum_{\substack{v \in K_u \\ r_{vi} \neq \emptyset}} s_{uv} r_{vi}, \quad R = \sum_{\substack{v \in K_u \\ r_{vi} \neq \emptyset}} |s_{uv}|$$

La clase *Similarity* engloba todas las posibles formas de calcular los coeficientes de similitud entre usuarios. Éstos se guardarán en la matriz de datos *matrix*, de tal forma que el elemento i, j de la matriz corresponde al coeficiente de similitud entre los usuarios i y j. Para las similitudes implementadas, estas matrices son simétricas, sin embargo se podrían implementar otras similitudes no simétricas (como la divergencia de Kullback-Leibler [14]), y por tanto, guardamos la matriz completa en lugar del triángulo superior.

Para evitar que se considere al propio usuario como el vecino más cercano, reduciendo entonces el número de vecinos efectivos a la hora de predecir ratings, establecemos la diagonal de la matriz de similitudes como un vector de ceros.

Hemos implementado las métricas de similitud más comúnmente utilizadas a la hora de realizar un sistema de recomendación basado en vecindarios [13], la similitud del coseno y la de Pearson, que se calculan de la siguiente manera:

Sean u, v dos usuarios. Sea J_u el conjunto de objetos puntuados por u, y J_v el conjunto de objetos puntuados por v. Sea J_{uv} el conjunto de objetos que han sido puntuados tanto por u como por v. El coseno entre u y v se define como:

$$cos(u, v) = \frac{\sum\limits_{i \in J_{uv}} r_{ui} r_{vi}}{\sqrt{\sum\limits_{i \in J_u} r_{ui}^2 \sum\limits_{i \in J_v} r_{vi}^2}} \in [0, 1]$$

Y la similitud de Pearson se define como:

$$Pearson(u, v) = \frac{\sum\limits_{i \in J_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum\limits_{i \in J_{uv}} (r_{ui} - \bar{r}_u)^2 \sum\limits_{i \in J_{uv}} (r_{vi} - \bar{r}_v)^2}} \in [-1, 1]$$

Donde $\bar{r}_u$ indica la media de las valoraciones del usuario u.

Las clases *Mahalanobis* y *MahalanobisUsers* se encargan de calcular la similitud basándose en la distancia de Mahalanobis. Los detalles de su implementación y la diferencia entre ambas están especificadas en la siguiente sección.

Cuando llamamos al método *compute*, se utiliza la matriz de valoraciones dada por la clase Ratings para calcular la matriz de similitudes completa, en caso de que nunca se haya calculado antes para dicho conjunto de entrenamiento; y posteriormente se guarda en disco para ahorrar tiempo de computación, de tal forma que en caso de que ya se haya calculado antes simplemente se lee desde el archivo.

Por último, la clase *Evaluation* es implementada por las distintas métricas de error utilizadas para medir la precisión de los distintos algoritmos o, en este caso, las distintas similitudes implementadas para el algoritmo basado en vecindarios implementado. Los detalles sobre estas métricas los proporcionaremos en el siguiente capítulo.

4.2.1. Consideraciones sobre la distancia de Mahalanobis

Recordemos en primer lugar la fórmula de la distancia de Mahalanobis entre dos vectores aleatorios:

$$d(x_1, x_2) = \sqrt{(x_1 - x_2)^t \Sigma^{-1} (x_1 - x_2)} \tag{4.1}$$

Siendo Σ la *matriz de covarianzas* del conjunto de datos.

Observando la fórmula, surgen varias cuestiones respecto a la forma en la que podemos aplicar esta distancia en el conjunto de datos pertinente:

- ¿Qué identificamos como el vector aleatorio para calcular la distancia?
- ¿Cuál es la matriz del conjunto de datos?
- ¿Cómo invertimos la potencialmente grande matriz de covarianzas?

En primer lugar, la selección del vector aleatorio parece sencilla en un primer momento: puesto que nos interesa calcular la similitud entre dos usuarios, elegimos la parte de los datos que identifica a cada uno: su vector de valoraciones. Sin embargo, los usuarios no han puntuado el mismo número de películas, y por supuesto, no han puntuado exactamente las mismas películas. Esto nos plantea otra duda dentro de la pregunta que estamos respondiendo, para la cual hemos pensado en dos aproximaciones:

- Tomar todo el vector de puntuaciones de cada usuario, incluyendo aquellos objetos que aún no ha puntuado.
- Tomar únicamente las valoraciones comunes a los dos usuarios, es decir, tomar únicamente las valoraciones para las cuales ambos usuarios hayan dado una puntuación.

Sean a, b los índices de los usuarios para los cuales estamos calculando su distancia de Mahalanobis. Sea J el conjunto total de objetos. Sea J_{ab} el conjunto de objetos que han sido puntuados tanto por a como por b.

Para cada una de estas opciones, la respuesta sobre qué matriz de datos tomar debe ser distinta. Esto es debido a la cuestión de las dimensiones que deben tener los vectores y la matriz de covarianzas para poder calcular la distancia de Mahalanobis. Atendamos de nuevo a la fórmula:

$$d(x_a, x_b) = \sqrt{(x_a - x_b)^t \Sigma^{-1} (x_a - x_b)}$$

La dimensión de la matriz Σ dependerá del tamaño de los vectores x_a, x_b que hayamos elegido. Vemos que en el primer caso, la dimensión de los vectores de usuario será de $|J|$, por tanto la matriz Σ debe tener dimensión $|J| \times |J|$ para que el producto matricial pueda realizarse. En el segundo caso, como la dimensión de los vectores de usuario será de $|J_{ab}|$, la matriz de covarianzas debe tener dimensión $|J_{ab}| \times |J_{ab}|$. Es decir, a la hora de calcular las covarianzas, debemos calcularlas tomando como variables aleatorias los distintos objetos, y no los usuarios (que simplemente son un vector de observaciones para cada variable aleatoria).

Hechas estas consideraciones, podemos establecer qué tomaremos como matriz de datos en cada caso. Para la primera aproximación la respuesta podría ser la matriz U completa, donde U_{ij} será la puntuación dada por el usuario i al objeto j. Sin embargo, habría que determinar qué valor tendrán las puntuaciones para las que el usuario no ha dado valoración, ya que estos datos se usarán tanto para calcular la matriz de covarianzas como el vector que identifica a los usuarios para los cuales estamos calculando la distancia. Poner un 0 en dichas puntuaciones introduciría demasiados datos incorrectos para calcular las distancias, ya que estaríamos estableciendo que el usuario daría la puntuación mínima a todos los objetos que no ha puntuado. Por tanto, pensamos en dos posibles alternativas: asignar a las puntuaciones vacías el valor medio de puntuaciones para el usuario al que corresponde dicha puntuación (clase *MahalanobisUsers* en figura 4.1), o asignarles el valor medio de puntuaciones para el objeto correspondiente (clase *Mahalanobis* en figura 4.1). Ambas alternativas han funcionado, sin embargo, creemos que la opción de introducir las puntuaciones medias de acuerdo a los objetos introducirá menos ruido a la hora de realizar las recomendaciones, ya que estamos calculando la matriz de covarianzas tomando como variables aleatorias los objetos, y no los usuarios, tal y como hemos dicho anteriormente.

Para el segundo caso (tomar sólo las valoraciones comunes a los usuarios), podemos elegir como matriz de datos U' de dimensión $N \times |J_{ab}|$, en la que sólo cogemos las columnas correspondientes a los objetos que han puntuado tanto a como b. Sin embargo, vemos que para cada par de usuarios que elegimos, tendríamos que calcular una matriz de covarianzas distinta y posteriormente invertirla, y además volvemos a tener el problema de los valores no puntuados por los usuarios. Debido a todas estas incertidumbres y a la potencial complejidad computacional, hemos decidido no implementar esta alternativa. Otra opción sería tomar como matriz de datos U'' de dimensión $2 \times |J_{ab}|$ con únicamente las valoraciones de los objetos que los dos usuarios para los que estamos calculando la distancia han puntuado. En esta opción no tenemos el problema de los objetos sin

valorar, y aunque tenemos que calcular una matriz de covarianzas e invertirla posteriormente para cada par de usuarios, su menor tamaño alivia parcialmente esta complejidad computacional.

$$U = \begin{array}{c} \begin{matrix} i_1 & i_2 & i_3 & i_4 & i_5 \end{matrix} \\ \begin{pmatrix} 2 & \_ & 5 & 1 & \_ \\ \_ & 3 & \_ & 4 & 2 \\ 4 & 3 & 3 & \_ & 3 \\ 1 & 4 & 4 & 5 & 3 \\ \_ & \_ & 4 & 1 & 4 \end{pmatrix} \end{array} \begin{matrix} u_1 \\ u_a \\ u_3 \\ u_b \\ u_5 \end{matrix} \quad U' = \begin{array}{c} \begin{matrix} i_2 & i_4 & i_5 \end{matrix} \\ \begin{pmatrix} \_ & 1 & \_ \\ 3 & 4 & 2 \\ 3 & \_ & 3 \\ 4 & 5 & 3 \\ \_ & 1 & 4 \end{pmatrix} \end{array} \begin{matrix} u_1 \\ u_a \\ u_3 \\ u_b \\ u_5 \end{matrix} \quad U'' = \begin{array}{c} \begin{matrix} i_2 & i_4 & i_5 \end{matrix} \\ \begin{pmatrix} 3 & 4 & 2 \\ 4 & 5 & 3 \end{pmatrix} \end{array} \begin{matrix} u_a \\ u_b \end{matrix}$$

Figura 4.2: Matrices de datos utilizadas en cada caso

Llegados a este punto nos quedan dos posibles alternativas de implementación siendo necesario decidir cómo invertir la matriz de covarianzas. En un primer intento, intentamos invertir la matriz de covarianzas tal cual en ambas opciones, sin embargo vimos que la matriz de covarianzas resultante no era invertible. Por tanto, decidimos calcular la matriz pseudoinversa utilizando descomposición en valores singulares (SVD), que es la mayor aproximación a la inversa para matrices no invertibles. Con esta opción, obtuvimos buenos resultados usando la matriz completa de datos U, sin embargo, utilizando la matriz recortada U'', su pequeño tamaño (en ocasiones tan pequeño que no era posible ni siquiera calcular la matriz de covarianzas) ocasionó demasiados problemas a la hora de calcular la distancia de Mahalanobis, ya que a la hora de calcular la pseudoinversa obtuvimos datos que provocaban que la fórmula de la distancia proporcionara valores incorrectos (como distancias negativas). Por esta razón, rechazamos la utilización de esta alternativa.

En conclusión, después de valorar las posibles decisiones de diseño y analizar las distintas opciones vistas a lo largo de este capítulo, la alternativa para implementar la distancia de Mahalanobis empleada tiene los siguientes pasos:

1. Cambiar la matriz de datos para que cada objeto que no haya puntuado cierto usuario tenga como valoración la media de las puntuaciones de dicho usuario o de dicho objeto.

2. Usar esta matriz para calcular la matriz de covarianzas entre objetos

3. Calcular la pseudoinversa de dicha matriz mediante el método de descomposición en valores singulares (SVD)

4. Para cada par de usuarios, calcular su distancia de Mahalanobis utilizando la fórmula (4.1).

Para calcular la covarianza se ha utilizado el método *cov* de la librería *NumPy*. De igual forma, para calcular la pseudoinversa se ha utilizado el método *pinv* del paquete *linalg* de la misma librería.

5

Experimentos Realizados y Resultados

En este capítulo describimos el conjunto de datos utilizados, así como los experimentos prácticos realizados con el sistema descrito en el capítulo anterior.

5.1. Descripción y análisis del conjunto de datos

A la hora de realizar un sistema de recomendación, el primer paso es recopilar todos los datos de los usuarios del sistema que puedan resultar pertinentes para realizar las recomendaciones. Usualmente, estos datos consisten en una matriz de datos donde el elemento en la fila i y columna j es la valoración dada por el usuario i al objeto j. Estas valoraciones pueden ser dadas por el usuario de forma implícita o explícita, por ejemplo [9]:

- Valoraciones implícitas unitarias, tales como "el usuario ha mirado este objeto" o "el usuario ha comprado este objeto".
- Valoraciones binarias, tales como cuando se pregunta al usuario si el objeto es bueno o malo.
- Valoraciones ordenadas, como preguntar al usuario si está "muy de acuerdo, de acuerdo, indiferente, en desacuerdo o totalmente en desacuerdo", seleccionando el término que está más cercano a su opinión respecto al objeto.
- Valoraciones numéricas, tales como valorar un objeto en un rango entre 1 y 5 estrellas.

Además de las valoraciones, es habitual recopilar información social como amistades entre usuarios, comentarios, información contextual como género, lugar de procedencia o edad, y datos temporales, es decir, en qué momento se realizó la valoración.

Para la realización de este trabajo, hemos utilizado los datos disponibles del sitio *MovieLens* [15]. *MovieLens* es un sistema de recomendación de películas creado por el grupo de investigadores *GroupLens Research* de la Universidad de Minessota. Este sistema, aparte de proporcionar información para los usuarios, recopila los datos anónimos de todos los usuarios que se utilizan para crear conjuntos de datos útiles para la investigación y la docencia.

MovieLens dispone de varios tamaños de conjuntos de datos disponibles, útiles para varias aplicaciones. En nuestro caso, debido a la baja capacidad de computación de los equipos utilizados durante el desarrollo del trabajo, hemos optado por escoger el conjunto *MovieLens 100K*. Este conjunto de datos comprende 100000 valoraciones hechas por 943 usuarios sobre 1682 películas. Además garantiza usuarios que hayan realizado al menos 20 valoraciones.

La parte principal de estos datos es el conjunto de valoraciones, en un archivo donde se encuentran filas que contienen el identificador del usuario, el identificador del objeto y la valoración que dicho usuario ha dado a dicho objeto, junto a una etiqueta de tiempo que indica cuándo se realizó la valoración. Estos datos también se encuentran separados en 5 particiones de los datos, divididos en conjuntos de entrenamiento y conjuntos de test, para entrenar el recomendador con los datos de entrenamiento y probar su precisión con los conjuntos de test. Los 5 conjuntos de test son disjuntos entre sí y comprenden cada uno el 20 % del total de los datos, haciéndolos ideales para realizar una valoración cruzada de 5 iteraciones.

Además de las valoraciones, el conjunto de datos contiene información contextual y demográfica que puede ser interesante para intentar mejorar la calidad de las recomendaciones. Respecto a las películas, se incluye un archivo que asocia cada identificador de película con su título, año de lanzamiento y los géneros en los que se incluye. Respecto a los usuarios, incluye una asociación de cada identificador con su edad, género, ocupación y código ZIP (equivalente estadounidense al código postal).

Puesto que en nuestro caso sólo hemos utilizado los datos de valoraciones de los usuarios, hemos realizado un análisis de dicho conjunto de datos, ya que queríamos comprender la información con la cual hemos trabajado. En concreto, hemos querido analizar la cantidad de valoraciones dadas para cada posible valor (figura 5.1), y además el número de usuarios que han realizado un cierto número de valoraciones (figura 5.2), de manera que se pueda realizar un filtrado del conjunto de datos en caso de que computacionalmente el coste sea demasiado elevado para los equipos con los que podemos trabajar.

En la figura 5.1, cada barra representa el número de valoraciones que se han dado para cada valor. En la figura 5.2, cada barra representa el número de usuarios que han realizado una cierta cantidad de valoraciones. Este gráfico está agrupado por clases, de 10 en 10 valoraciones, para facilitar su legibilidad.

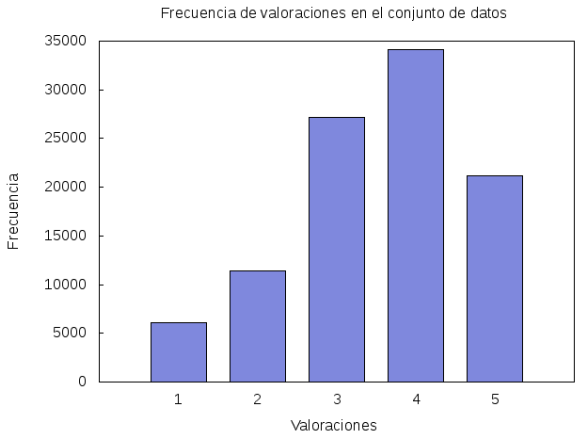

Figura 5.1: Numero de valoraciones por valor

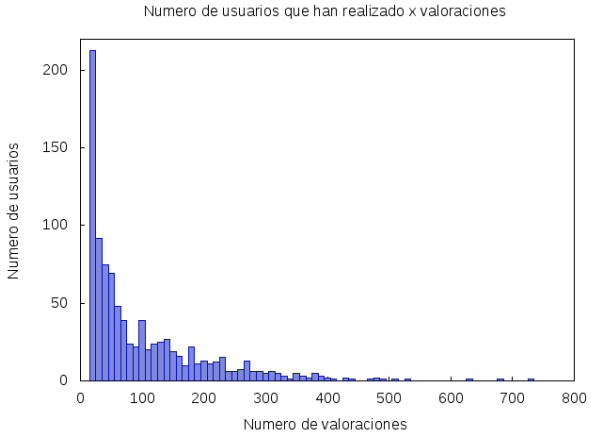

Figura 5.2: Numero de usuarios que han realizado un número de valoraciones

5.2. Medidas de error utilizadas

Las medidas de error son utilizadas comúnmente en la literatura para medir la calidad de los distintos métodos utilizados para implementar un sistema de recomendación [13]. Los pasos comunes a seguir para medir la precisión de las recomendaciones son:

- En primer lugar, se separa el conjunto de datos en dos partes. Una de ellas, llamada el *conjunto de entrenamiento*, se utiliza para calcular la información necesaria para realizar las recomendaciones, como hallar los distintos factores de un modelo planteado o, como es nuestro caso, calcular los factores de similitud entre los distintos usuarios.

- Una vez entrenado el sistema, se utiliza la otra parte del conjunto de datos, llamada *conjunto de test*, para medir su precisión. Como en los datos de este conjunto tenemos la valoración real realizada por el usuario, usamos este dato para calcular el error que comete nuestro recomendador a la hora de predecir esa valoración. Procesando todos los datos del conjunto de test, podemos utilizar una de las métricas de error que detallaremos más adelante para estimar la calidad del recomendador.

- Para no basar nuestra estimación en un único conjunto de datos se usa la técnica de *validación cruzada*, que consiste en realizar varias particiones distintas del conjunto de datos en conjuntos de entrenamiento y test, para luego promediar los errores obtenidos por cada partición.

El error en las recomendaciones se puede medir de diversas formas, las más comunes son el error absoluto medio o *Mean Absolute Error* (MAE), y la raíz del error cuadrático medio, o *Root Mean Squared Error* (RMSE), que se calculan de la siguiente forma:

Sea C el conjunto de datos, y sean C_{train}, C_{test} las particiones en conjuntos de entrenamiento y test, respectivamente. Las medidas del error serían:

$$MAE = \frac{1}{|C_{test}|} \sum_{r_{ui} \in C_{test}} |\hat{r}_{ui} - r_{ui}|$$

$$RMSE = \sqrt{\frac{1}{|C_{test}|} \sum_{r_{ui} \in C_{test}} (\hat{r}_{ui} - r_{ui})^2}$$

5.3. Comparativa de errores

Nuestro análisis se centra en la comparación entre la nueva forma de medir la similitud entre usuarios, utilizando la distancia de Mahalanobis, y las medidas utilizadas tradicionalmente, esto es, la distancia basada en el coseno y el coeficiente de correlación de Pearson.

Hemos realizado medidas del error en la recomendación tanto mediante RMSE como con MAE para el método basado en vecindarios con k usuarios más próximos. Para realizar las medidas, hemos utilizado la validación cruzada de 5 iteraciones con el conjunto de datos *MovieLens-100K*.

Esta medida se ha realizado variando el número de vecinos utilizados para realizar las predicciones, comenzando con 10 y terminando con 650, ya que utilizando más, el tiempo de computación era excesivamente alto.

En las figuras 5.3 y 5.4 están representados los errores respecto al número de vecinos. Hemos comparado las dos versiones de la implementación de la distancia de Mahalanobis propuestas, tal y como explicamos en el capítulo 4: una sustituyendo las puntuaciones no realizadas por la media de las demás puntuaciones del usuario y otra sustituyéndolas con la media de las valoraciones dadas al objeto. Adicionalmente, hemos querido comparar la precisión de este método de cálculo de similitudes con las medidas utilizadas tradicionalmente: la distancia del coseno y la correlación de Pearson.

Los resultados muestran que la precisión de la distancia del coseno es significativamente superior a las demás para este conjunto de datos. En cuanto a la comparación entre las dos implementaciones propuestas de la distancia de Mahalanobis, observamos que la propuesta de utilizar las puntuaciones medias de los usuarios obtiene resultados

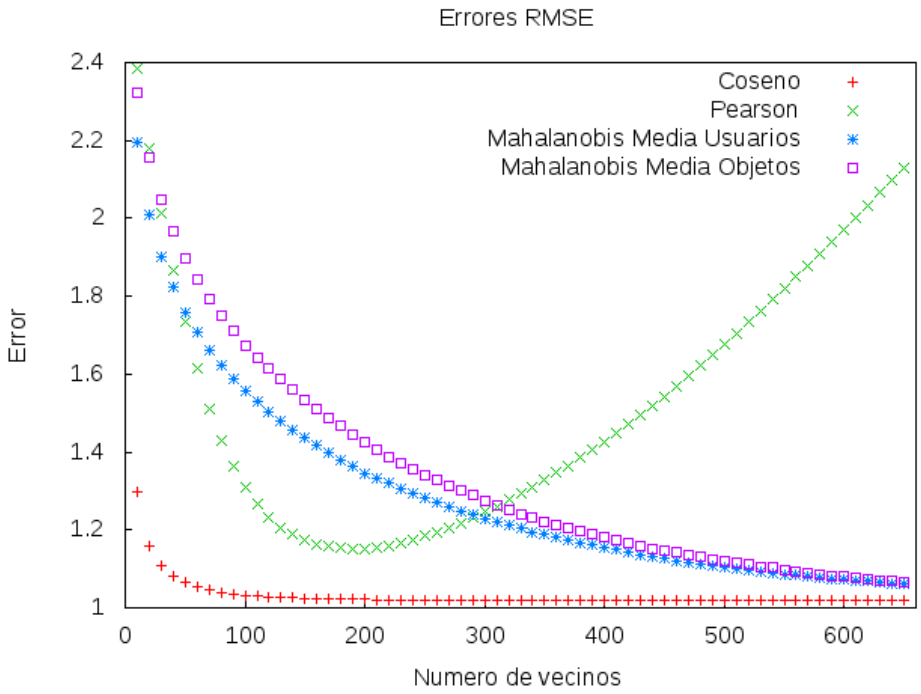

Figura 5.3: RMSE para las distintas similitudes. En el gráfico, Mahalanobis Media Usuarios representa la aproximación según la cual sustituimos las puntuaciones vacías por la media de las puntuaciones del usuario, y Mahalanobis Media Objetos aquella en la que las sustituimos por la puntuación media para el objeto

ligeramente mejores, al contrario de lo que esperábamos en un principio. Es necesario mencionar el comportamiento extraño del coeficiente de correlación de Pearson, ya que aunque en otros sistemas se ha demostrado que tiene una efectividad tan grande o incluso mayor que la del coseno [13], en este caso hemos obtenido unos resultados bastante por debajo de éste. Esto es debido al pequeño tamaño del conjunto de datos, lo que provoca que en ocasiones las correlaciones calculadas sean negativas, y esto afecta en gran medida a la calidad de la predicción ya que el algoritmo utilizado no tiene un buen comportamiento ante coeficientes de similitud negativos.

5.4. Eficiencia del sistema

De igual forma, también cabe preguntar si el nuevo sistema de recomendación planteado es más eficiente, esto es, mejora el tiempo empleado en calcular las predicciones, de manera que pueda resultar de interés a la hora de realizar recomendaciones en tiempo real, donde las exigencias temporales son mucho mayores.

Puesto que, como dijimos en el capitulo anterior, el sistema está diseñado desde un punto de vista de utilidad para realizar las distintas medidas, los tiempos de cada ejecución, donde se calcula una gran cantidad de datos antes de realizar cada recomendación (más datos de los que serían necesarios para realizar una recomendación a un único usua-

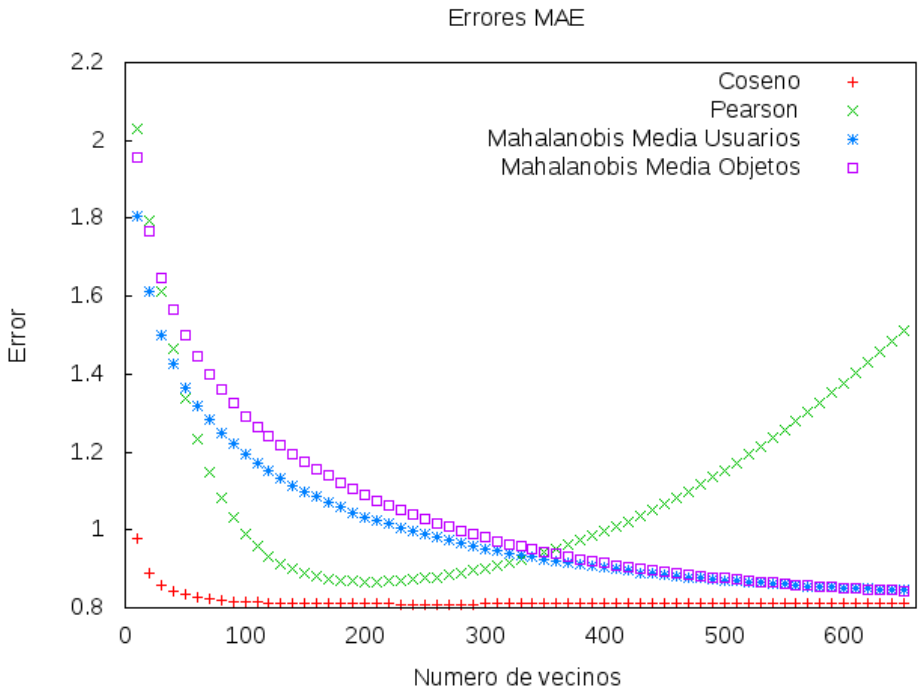

Figura 5.4: MAE para las distintas similitudes

rio), no se corresponden con el tiempo que emplearía este proceso en un sistema orientado al uso en la vida real.

Debido a esto, para realizar las medidas de tiempo, hemos querido tener en cuenta dos momentos en la ejecución: el cálculo de la similitud entre dos usuarios, y el cálculo de las similitudes entre un usuario y todos los demás.

Tipo	Mahalanobis Usuarios	Mahalanobis Objetos	Coseno	Pearson
s_{ij}	9.59786	10.96935	0.00162	0.00198
$s_{i:}$	12.92469	12.90806	1.44822	1.47401

Cuadro 5.1: Tiempos de cálculo de las similitudes, en segundos

En la tabla 5.1 se expresan los tiempos que tarda el sistema en calcular la similitud entre dos usuarios y entre un usuario y el resto, respectivamente, para cada similitud implementada.

En estos resultados notamos que los tiempos de procesamiento para las similitudes con Mahalanobis son mucho más altos. Esto es debido a que en esta estimación del tiempo hemos incluido las asignaciones de puntuaciones medias en las posiciones vacías, el cálculo de la matriz de covarianzas y su inversa, ya que es tiempo que emplea el sistema. Sin embargo, como este cálculo sólo se realiza una vez a lo largo de todo el procesamiento de la matriz de similitudes, hemos querido desglosar este tiempo para facilitar su análisis:

Medias Usuarios	Medias Objetos	Σ^{-1}
1.76146	2.16132	8.03152

Cuadro 5.2: Tiempos para las distintas funciones empleadas en Mahalanobis

Como observamos en la tabla 5.2, la función que más tiempo emplea es el cálculo de la matriz de covarianzas y posteriormente su inversa. Sin embargo, este cálculo, junto con la asignación de las puntuaciones medias, sólo se realizan una vez para todo el conjunto de datos, por lo que a la hora de calcular la matriz de similitudes al completo este tiempo no es tan significativo.

6

Conclusiones y trabajo futuro

Este Trabajo de Fin de Grado sirve como conclusión de los estudios realizados en el Doble Grado en Ingeniería Informática y Matemáticas, y a tal efecto, se ha realizado una labor de investigación y de desarrollo que comprende competencias de ambos ámbitos.

La idea inicial era utilizar una distancia poco usual, como la distancia de Mahalanobis, para realizar un cálculo de similitudes entre objetos, por ejemplo usuarios en un sistema de recomendación. Pensamos que sería interesante estudiarlo debido a que esta distancia, que utiliza información sobre la correlación entre todos los objetos del conjunto de datos a la hora de calcular la distancia entre dos objetos, podría aportar nuevas perspectivas sobre posibles formas de modelar las relaciones entre usuarios de un sistema de recomendación.

A tal efecto, realizamos una revisión previa sobre las aplicaciones que podría tener esta distancia en distintos ámbitos, y sobre los sistemas de recomendación, con el objetivo de incorporar esta distancia a su uso.

En cuanto al desarrollo, hemos implementado una aplicación para obtener medidas de la eficacia y eficiencia de este nuevo sistema planteado. Para llevar esto a cabo, se han usado diversas técnicas, como la programación de distintos algoritmos y métricas, la utilización del lenguaje *Python*, poco explorado en el Doble Grado, y el manejo de un conjunto considerable de resultados para elaborar gráficas y tablas.

A la vista de los resultados y del resto del trabajo, las conclusiones más importantes que podemos extraer son las siguientes:

- **Desarrollo del sistema en Python.** Debido a la existencia de librerías de sistemas de recomendación desarrolladas en lenguajes populares como *Java* o *C#*, las cuales son ampliamente utilizadas en artículos de investigación sobre este tema, decidimos utilizar un lenguaje sencillo, con grandes facilidades para el procesamiento

de datos, y que no tuviera una herramienta ampliamente utilizada implementada en él. Por ello, decidimos usar *Python*, que, junto a la librería *SciPy*, la cual incluye el conjunto de funciones y utilidades *NumPy*, facilita sobremanera el procesamiento de grandes cantidades de datos, obteniendo además un código más pequeño y legible. Debido a la buena experiencia, tanto en eficiencia como en facilidad de uso, que hemos tenido con este lenguaje, consideramos un gran acierto su elección.

- **Importancia de la manera en la que representamos los usuarios.** De igual forma que utilizando el coeficiente de correlación de Pearson, con la distancia de Mahalanobis los usuarios representan vectores aleatorios. Sin embargo, a diferencia del coeficiente de correlación, en la distancia de Mahalanobis necesitamos tener en cuenta el resto de los datos, y puesto que no se tienen todos los datos posibles (los usuarios no puntúan todos los objetos disponibles), la elección de los vectores y la matriz de datos utilizados resulta crucial a la hora de abordar el problema.

- **Resultados por debajo de lo esperado** Aunque los resultados obtenidos para nuestra distancia superan en gran medida a los calculados para el coeficiente de correlación de Pearson, este resultado no es especialmente significativo, ya que como hemos explicado anteriormente, la correlación de Pearson tiene valores negativos debido al relativamente pequeño tamaño del conjunto de datos, y esto resulta muy problemático a la hora de calcular las predicciones basándonos en vecindarios. Sin embargo, fijándonos en la otra medida estándar, el coseno, los resultados dejan bastante que desear, tanto en eficacia como en eficiencia. Sin embargo, creemos que los resultados pueden llegar a mejorarse, tomando nuevas aproximaciones descritas en la siguiente sección, pero no podemos aventurarnos a decir que llegarán a superar a las medidas utilizadas tradicionalmente

6.1. Trabajos futuros

Tras realizar este trabajo, quedan planteadas ciertas incógnitas que podrían explorarse en trabajos posteriores, las cuales se detallan a continuación:

- **Realizar un sistema de recomendación colaborativo basado en objetos.** De la misma forma en la que se calculan similitudes entre usuarios y se hace un promedio ponderado entre ellos para predecir la valoración que un usuario dará a un objeto, se puede realizar un promedio entre los objetos más similares entre los puntuados por el usuario para el que estamos calculando la predicción. Esta vertiente podría mejorar la precisión de las recomendaciones, aunque posiblemente pierda en cuanto a novedad de la recomendación dada. Con todo esto, pensamos que su estudio sería interesante.

- **Probar distintos conjuntos de datos.** En este trabajo se ha utilizado un único conjunto de datos, *MovieLens-100K*, para realizar los experimentos. Entre los varios

factores de los que depende la eficacia de un sistema de recomendación, encontramos al conjunto de datos utilizado, ya que distintas formas de dar las valoraciones, distintos tamaños del conjunto, o distintos tipos de objetos, pueden ser ajustados mejor por un determinado modelo. Con el objetivo de encontrar un tipo de datos para el que la distancia analizada modele mejor la relación entre los usuarios, pensamos que sería de gran interés analizar conjuntos de datos de diferentes tipos.

- **Implementar la distancia en librerías existentes.** Hemos mencionado anteriormente nuestro intento de implementar la distancia analizada en librerías de sistemas de recomendación existentes y ampliamente utilizadas en varias investigaciones sobre el tema. Debido a la complejidad adicional que suponía esta tarea, decidimos no seguir por esa línea de trabajo. Sin embargo, es de esperar que dichas librerías trabajen de manera más eficiente que nuestro sistema implementado. Además, cabe la posibilidad de utilizar la distancia implementada para algoritmos distintos al utilizado en este trabajo, que usen la similitud entre usuarios y/o objetos de maneras distintas para realizar las predicciones. Debido a esto, pensamos que se trata de una línea de trabajo interesante que puede ser llevada a cabo si se dispone de tiempo para investigar la librería pertinente o si se tiene conocimiento previo del uso y funcionamiento interno de dicha librería.

- **Incorporar datos contextuales al modelo.** En el sistema implementado, la única información tenida en cuenta a la hora de calcular las similitudes entre los distintos usuarios han sido las distintas valoraciones que los usuarios han dado a los objetos. Sin embargo, disponemos de una mayor cantidad de datos que pueden ayudar a modelar este grado de similitud, como pueden ser la edad o el género, o factores relacionados con los objetos que valoran, como en este caso el momento en el que se realiza la valoración, o etiquetas de tipo asociadas a los objetos valorados, como en este caso, los géneros a los que pertenece cada película. Aunque en ocasiones el exceso de datos proporcionados al modelo puede causar sobreajuste, pensamos que esta línea de investigación podría mejorar los resultados obtenidos.

- **Realizar un procesamiento previo de los datos.** En ocasiones, el uso de los datos tal y como llegan del sistema puede llevar a resultados subóptimos, debido a que distintos factores como la media y la varianza pueden posicionar los datos de tal forma que la medida de similitud utilizada no pueda realizar el modelo de los datos correcto. A tal efecto, es común en los sistemas de recomendación realizar una *normalización* previa de los datos, cuyo objetivo es eliminar la parte menos representativa de éstos para ayudar a modelar las relaciones entre usuarios y objetos. Creemos que podría ser interesante investigar el efecto de las distintas formas de realizar la normalización de datos en los resultados obtenidos.

Bibliografía

[1] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, December 1992.

[2] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM.

[3] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2006.

[4] Juraj M. Cunderlik and Donald H. Burn. Switching the pooling similarity distances: Mahalanobis for euclidean. *Water Resources Research*, 42(3):n/a–n/a, 2006. W03409.

[5] Daniel Wolff and Tillman Weyde. *Combining Sources of Description for Approximating Music Similarity Ratings*, pages 114–124. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[6] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender Systems Handbook*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[7] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. MyMediaLite: A free recommender system library. In *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys 2011)*, 2011.

[8] Apache mahout. `https://mahout.apache.org/`. Ultimo acceso: 23/06/2017.

[9] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to Recommender Systems Handbook*, pages 1–35. Springer US, Boston, MA, 2011.

[10] Adi Ben-Israel and Thomas NE Greville. *Generalized inverses: theory and applications*, volume 15. Springer Science & Business Media, 2003.

[11] Carles M. Cuadras. Distancias estadísticas. *Estadística Española*, 30(119):295–378, 1989.

[12] R. De Maesschalck, D. Jouan-Rimbaud, and D.L. Massart. The mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50(1):1 – 18, 2000.

[13] Christian Desrosiers and George Karypis. *A Comprehensive Survey of Neighborhood-based Recommendation Methods*, pages 107–144. Springer US, Boston, MA, 2011.

[14] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern classification*, chapter Stochastic methods, pages 350–394. Pattern Classification and Scene Analysis: Pattern Classification. Wiley, 2001.

[15] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, December 2015.